

Problema 1

O método de Gram Schmidt para ortogonalização de vetores é um método iterativo que, dado um conjunto de vetores linearmente independentes, gera um conjunto de vetores ortogonais. O método é baseado na projeção de um vetor sobre os vetores anteriores.

Implementação

qr_GS.m

```
% Entradas:
% A - matriz (m x n)
% Saídas:
% Q = matriz (m x n) ortogonal
% R = matriz (n x n) triangular superior
function [Q,R] = qr_GS(A)
    [m, n] = size(A);

    % Inicializa matrizes
    Q = zeros(m,n);
    R = zeros(n);

    for j = 1 : n
        v = A(:,j); % j-ésima coluna de A

        % Obtém, por Gram-Schmidt, v o j-ésimo vetor de uma base ortogonal
        for i = 1 : j-1
            R(i,j) = dot(Q(:,i), A(:,j));
            v = v - R(i,j) * Q(:,i);
        end

        R(j,j) = norm(v,2);
        Q(:,j) = v / R(j,j); % j-ésimo vetor de uma base ortonormal
    end
end
```

Testes

A seguir estão algumas matrizes selecionadas para testar as funções implementadas neste trabalho.

$$A = \begin{pmatrix} 1 & 2 & 2 \\ -1 & 1 & 2 \\ -1 & 0 & 1 \\ 1 & 1 & 2 \end{pmatrix} \quad B = \begin{pmatrix} 16 & 2 & 3 & 13 \\ 5 & 11 & 10 & 8 \\ 9 & 7 & 6 & 12 \\ 4 & 14 & 15 & 1 \end{pmatrix} \quad C = \begin{pmatrix} 2 & 1 & 2 & 4 \\ 3 & 1 & 2 & 2 \end{pmatrix}$$

A é uma matriz ideal para ortogonalização (tanto que é um exemplo dado pelo Poole), pois contém vetores para uma base de um subespaço vetorial.

B é uma matriz mágica de ordem par, portanto é muito má condicionada (ela é quase singular), o que é interessante para os testes.

C é uma matriz retangular com mais colunas do que linhas, o que não é o caso esperado de acordo com a apresentação da fatoração QR (a que é feita no Poole), contudo é um cenário de prova para a implementação e interpretação do método.

Testando para o cenário ideal (a matriz A), temos:

```
>> [QCa, RCa] = qr_GS(A);

QCa =
    0.5    0.67082   -0.40825
   -0.5    0.67082         0
   -0.5    0.22361    0.40825
    0.5    0.22361    0.8165

RCa =
     2         1         0.5
     0    2.2361    3.3541
     0         0    1.2247
```

Considerando $A = QR$, para verificar a ortogonalidade de Q , calculamos $Q^T Q$ e para verificar a acurácia decomposição QR , calculamos $QR - A$.

```
>> QCa' * QCa
     1         0         0
     0         1   -2.7e-17
     0   -2.7e-17         1

>> QCa*RCa - A
     0     0     0
     0     0     0
     0     0     0
     0     0     0
```

Pode ser visto que a decomposição QR obtida foi muito boa. Q não é por muito pouco (o erro é irrelevante, tem grandeza 10^{-17}) a identidade, e a multiplicação de Q e R resulta em A .

Para as outras matrizes (B e C), temos:

Ambas são boas fatorações, afinal a multiplicação das matrizes resulta na matriz original, ou algo muito próximo disso. Vale ressaltar que a fatoração funciona para C (uma matriz com mais colunas do que linhas), mesmo que esvaziada do sentido.

```
>> QCb*RCb - B
     0     0     0     0
     0     0     0     0
     0     0     0     0
     0     0     0     0

>> QCc*RCc - C
   -2.2e-16     0     0     0
           0     0     0     0
```

Testando a ortogonalidade de B, vemos que o resultado não é o melhor, muitas entradas são muito próximas de zero, outras nem tanto (na ordem de 10^{-1}). Isso é uma consequência do mal condicionamento de B.

```
>> QCb' * QCb
      1 -2.7e-17 4.9e-16 0.55125
-2.7e-17 1 -5.5e-16 -0.25841
4.9e-16 -5.5e-16 1 -0.7925
0.55125 -0.25841 -0.7925 1
```

Para C, temos que avaliar algo diferente, afinal a matriz Q associada a ela não pode ser ortogonal, ela não é LI. Note que há um bloco que é a identidade, o que acontece devido ao fato de que os dois primeiros vetores são LI, se fossem outros, o bloco da identidade estaria em outra posição (ao menos é).

```
>> QCc' * QCc
      1 1.4e-15 0.00377 0.00377
1.4e-15 1 -0.99999 -0.99999
0.00377 -0.99999 1 1
0.00377 -0.99999 1 1

>> QCc(1:2, 1:2)' * QCc(1:2, 1:2)
      1 1.4e-15
1.4e-15 1
```